

FIG. 1

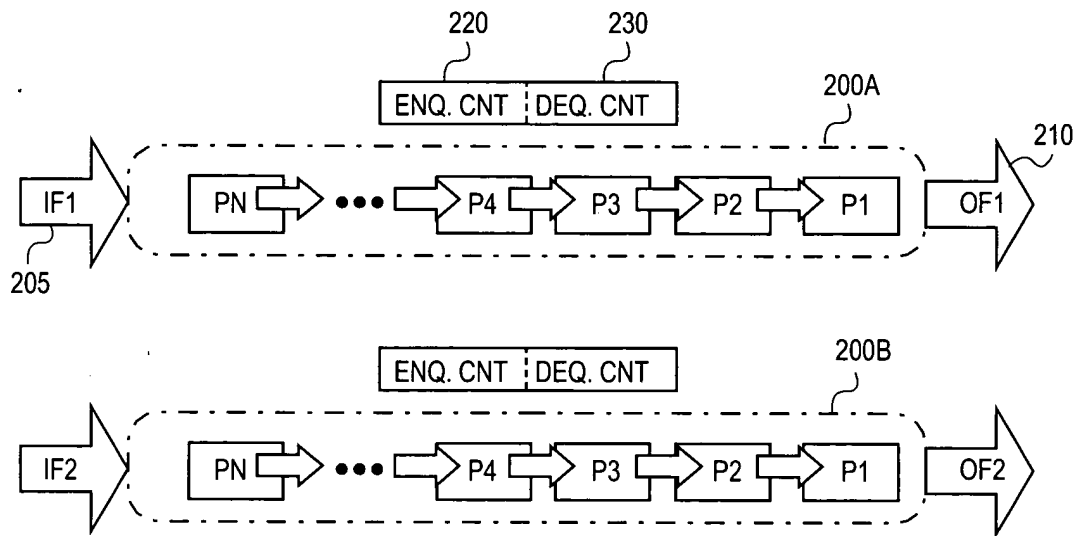


FIG. 2

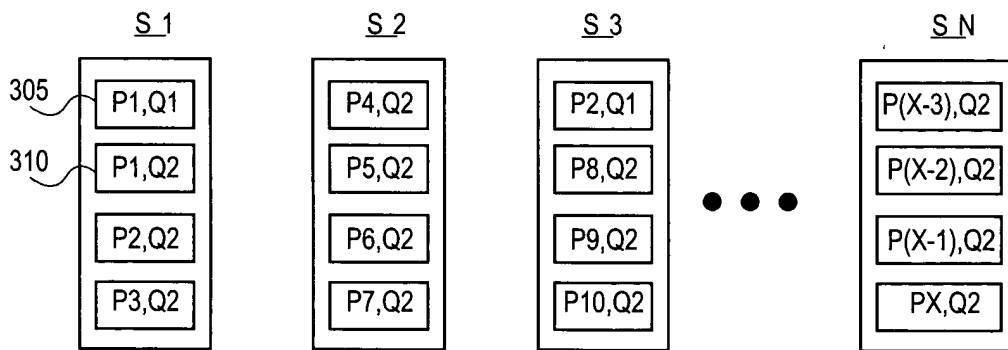


FIG. 3

300

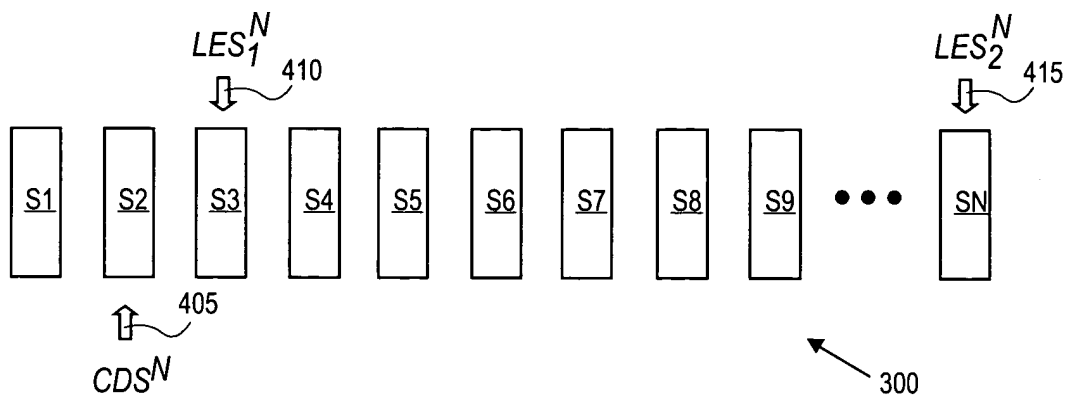


FIG. 4

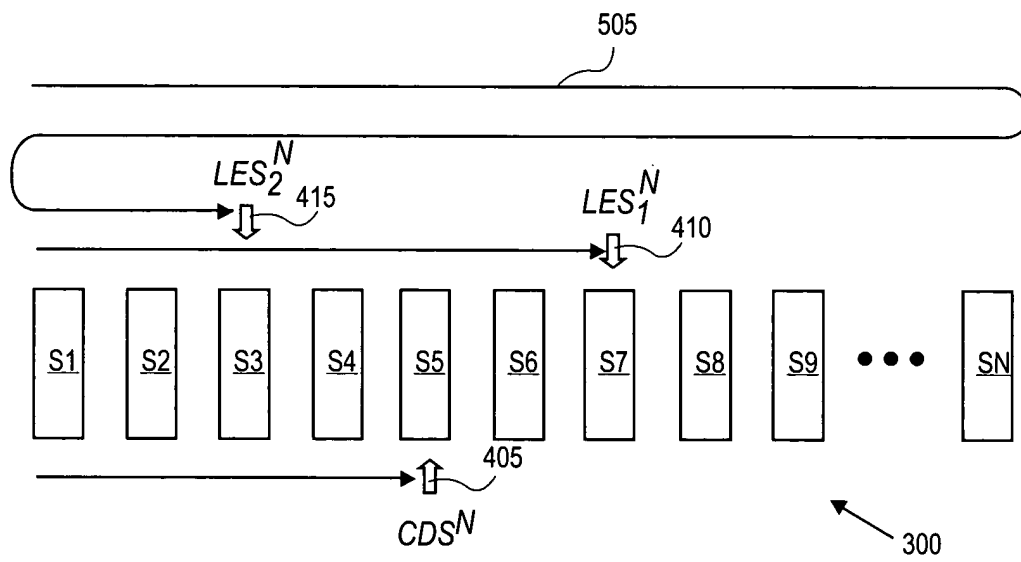


FIG. 5

600

610

```

enqueue( queue i)
{
    if (enqueue_count == dequeue_count) // check #1
    {
         $LES_i^N = CDS^N$ ;           // queue is empty
    }
    else if ((( $CDS^N - LES_i^N$ ) mod N) < M) // check #2
    {
         $LES_i^N = CDS^N$ ;           // queue is empty and dequeue count is
                                   // lagging
    }

     $LES_i^N = (LES_i^N + j)$  mod N; // calculate where to enqueue the packet
                                   // value j < M depends on queuing scheme
                                   // Note:  $LES_i^N$  may increase,  $CDS^N$  unchanged

    if ((( $CDR^N - LSR_i^N$ ) mod N) < M) // check #3
    {
        Drop packet                // queue has overflowed
         $LES_i^N = (LES_i^N - j)$  mod N; // reset  $LES_i$  to old value
    }
    else
    {
        Enqueue the packet
    }
}

```

605

```

while (1)
{
    enqueue(i);                    // call enqueue routine for queue
                                   // value  $LES_i$  may increase

    Perform some dequeues
    Perform enqueues on other queues
     $CDS^N = (CDS^N + k)$  mod N;      // where  $k \geq 0$ , k depends on how many rounds
                                   // have completed dequeues
                                   // Note:  $LES_i^N$  unchanged,  $CDS^N$  may increase
}

```

FIG. 6

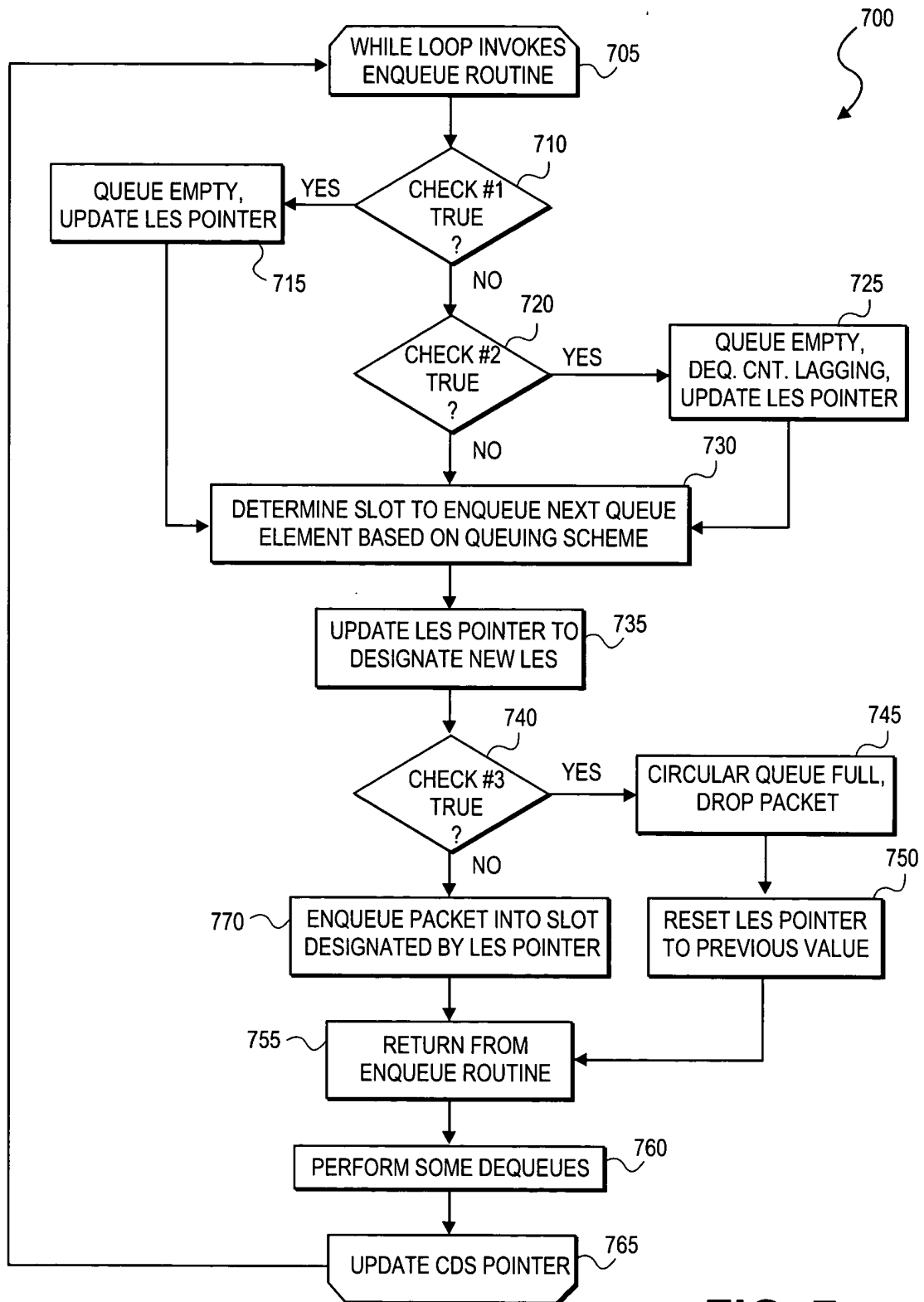


FIG. 7

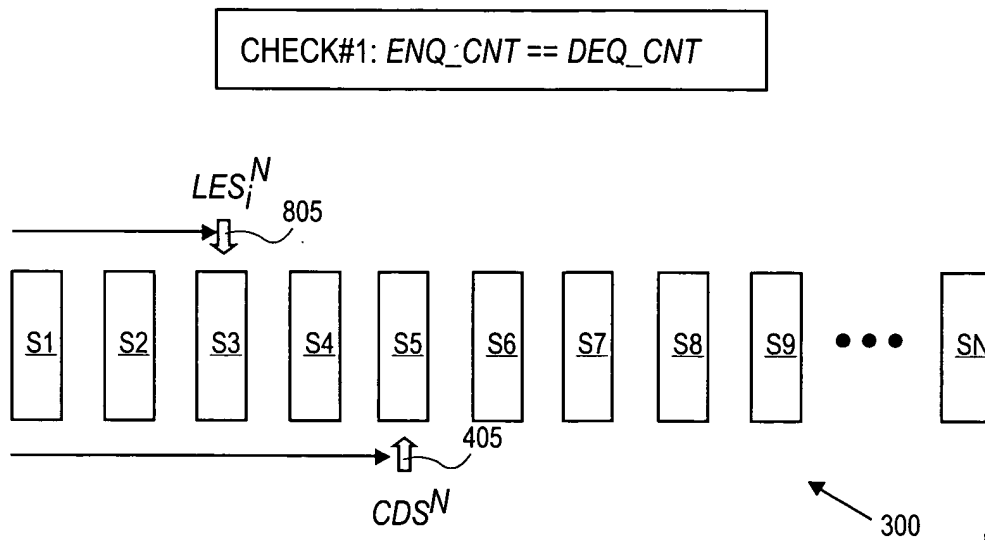


FIG. 8

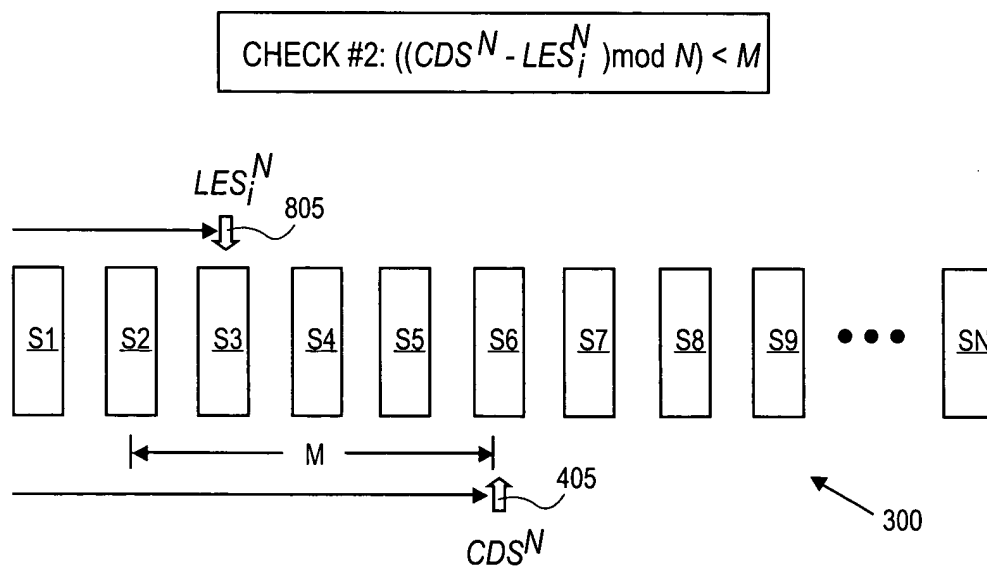


FIG. 9

$$\text{CHECK \#3: } ((CDS^N - LES_i^N) \bmod N) < M$$

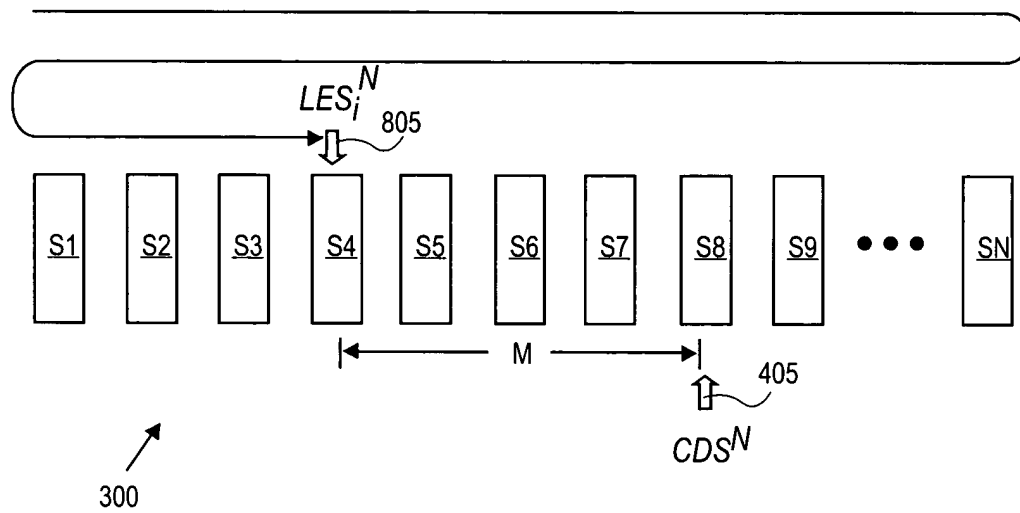


FIG. 10

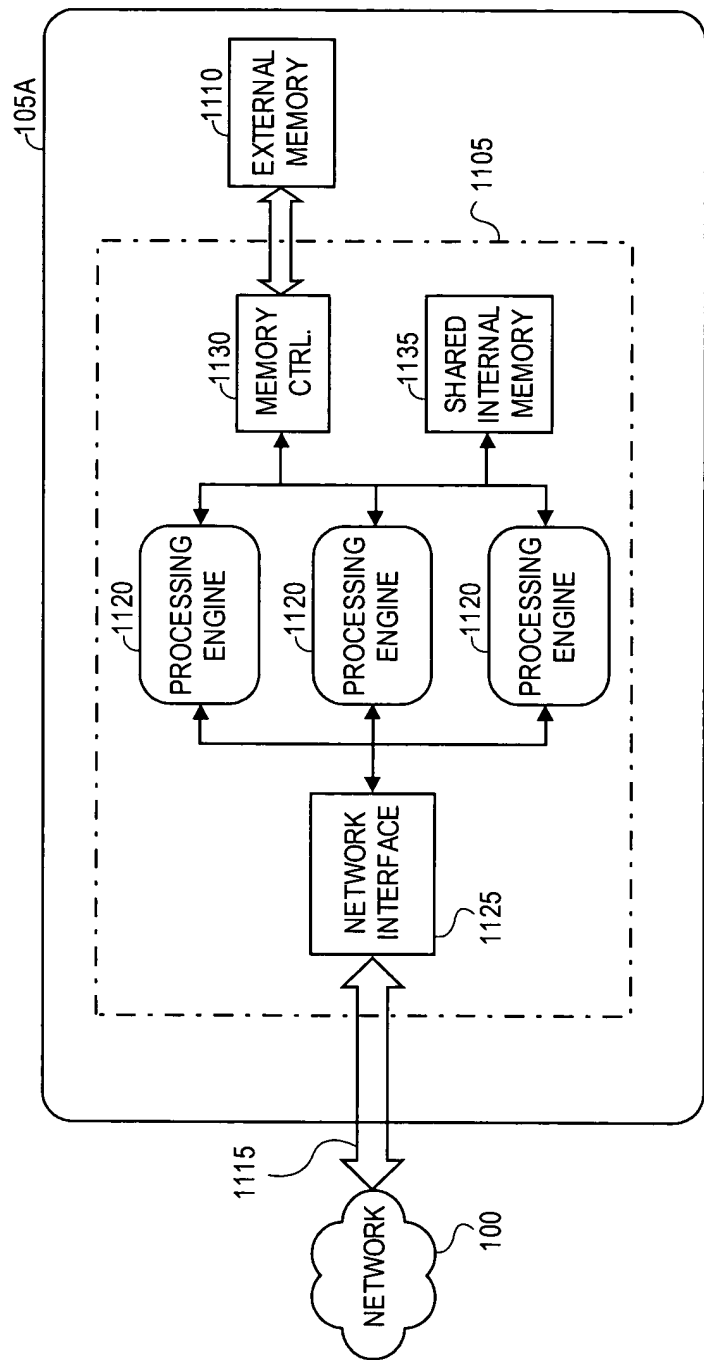


FIG. 11